MAC0323 - Relatório EP 1

Eduardo Brancher Urenha NUSP 8587409

Desenho dos Experimentos

Para fazer os experimentos, foi utilizada uma série de fatores de escala (de 256 a 1) que dividem o tamanho do texto. Desse modo, criamos desde tabelas de símbolos muito pequenas até uma que contém todas as palavras do texto, para que possamos observar bem os efeitos da variação do tamanho da tabela no tempo das operações. Além disso, o uso de fatores de escala grandes (levando a tabelas pequenas) é recomendado para executar o programa junto com o valgrind, de modo a minimizar o tempo de espera introduzido pela checagem da memória.

Os experimentos são encerrados quando o fator de escala se torna um fator mínimo, (por default 1) definido nos argumentos do programa, como explicado no arquivo Instruções de Uso. Notamos que, embora o número de palavras inseridas dobre em cada experimento (pois dividimos o fator por dois, logo percorremos duas vezes mais texto durante a construção da tabela), o tamanho reportado aqui representa a quantidade efetiva de palavras distintas, uma vez que palavras repetidas "entram no mesmo espaço de memória"e portanto não afetam o tamanho da tabela e duração das operações. Logo o tamanho reportado nos experimentos refere-se à quantidade de nós distintos na tabela. Todos os tempos são em segundos.

Cabe também notar que a ordenação de strings em UTF-8 de acordo com o C++ não está de acordo com a norma culta da língua portuguesa, uma vez que a string pá é maior que pi, por exemplo. Isso se deve ao fato de que o valor numérico do caractere á deve ser maior que i.

Resultados

1. Tabela de Hashing

   (a) inserção

tamanho	tempo médio(sec)
148	0.0000096343
258	0.0000069917
453	0.0000046361
826	0.0000054686
1397	0.0000060549
2278	0.0000036653
3762	0.0000052782
6151	0.0000038851
9523	0.0000051228

   (b) consulta

tamanho	tempo médio(sec)
148	0.0000040259
258	0.0000016787
453	0.0000016750
826	0.0000023392
1397	0.0000017696
2278	0.0000016980
3762	0.0000017188
6151	0.0000016921
9523	0.0000017782

(c) rankeamento

tamanho	tempo médio(sec)
148	0.0000253676
258	0.0000395250
453	0.0000683148
826	0.0001190049
1397	0.0002062647
2278	0.0004512911
3762	0.0005847614
6151	0.0010275050
9523	0.0015672752

(d) seleção

tamanho	tempo médio(sec)
148	0.0016845519
258	0.0056840537
453	0.0153916324
826	0.0505446059
1397	0.14028786679
2278	0.3369901891
3762	1.1942483406
6151	2.5160575584
9523	5.2359401267

(e) remoção

tamanho	tempo médio(sec)
148	0.0000050500
258	0.0000031037
453	0.0000018000
826	0.0000016098
1397	0.0000026490
2278	0.0000021030
3762	0.0000030040
6151	0.0000022327
9523	0.0000039139

Os experimentos parecem comprovar que as operações de inserção, remoção e consulta são praticamente constantes em tabelas de Hashing que tem o tamanho aproximado do texto e uma boa função de dispersão. Contudo, o ato de ordenar as chaves requer que olhemos todas as chaves linearmente, pois elas não são ordenadas, logo ele é linear. E como não guardamos o ranking das chaves, para selecionar uma chave com um dado rank precisamos calcular o rank (linear) de todas as chaves do vetor (linear). Logo a operação de seleção é quadrática em tabelas de hashing. Os experimentos parecem

confirmar ambas as hipóteses, pois aumentos entre 1 e 2 no tamanho do rankeamento levam a aumentos de tempo entre 1 e 2, e aumentos entre 1 e 2 no tamanho da seleção levam a aumentos de tempo entre 2 e 4.

2. Vetor Desordenado

   (a) inserção

tamanho	tempo médio(sec)
453	0.0000218935
826	0.0000386902
1397	0.0000629745
2278	0.0000720842
3762	0.0001226822
6151	0.0001926475
9523	0.0002295109

   (b) consulta

tamanho	tempo médio(sec)
453	0.0000208722
826	0.0000364676
1397	0.0000619098
2278	0.0000720901
3762	0.0001209554
6151	0.0001928000
9523	0.0002289842

   (c) rankeamento

tamanho	tempo médio(sec)
453	0.0000604352
826	0.0001092559
1397	0.0001846471
2278	0.0003059505
3762	0.0004954455
6151	0.0008279594
9523	0.0013328901

   (d) seleção

tamanho	tempo médio(sec)
453	0.0103813306
826	0.0306114392
1397	0.0927784137
2278	0.1709239000
3762	0.5034583871
6151	1.3210631337
9523	2.4395487495

   (e) remoção

tamanho	tempo médio(sec)
453	0.0000286907
826	0.0000470245
1397	0.0001173158
2278	0.0001173158
3762	0.0001744604
6151	0.0003158238
9523	0.0004505891

Podemos notar que toda operação é linear pois o tempo aumenta proporcionalmente ao tamanho, exceto por seleção que é quadrática, pois envolve duas operações lineares encadeadas. Nota-se que a tabela de hashing é vastamente superior ao vetor ordenado.

3. Vetor Ordenado

   (a) inserção

tamanho	tempo médio(sec)
453	0.0000569602
826	0.0001062157
1397	0.0001787176
2278	0.0003353396
3762	0.0005494347
6151	0.0009146475
9523	0.0014647158

   (b) consulta

tamanho	tempo médio(sec)
453	0.0000036639
826	0.0000047716
1397	0.0000046373
2278	0.0000046822
3762	0.0000047614
6151	0.0000050396
9523	0.0000061723

   (c) rankeamento

tamanho	tempo médio(sec)
453	0.0000035204
826	0.0000041716
1397	0.0000045863
2278	0.0000047792
3762	0.0000048792
6151	0.0000049416
9523	0.0000053218

   (d) seleção

tamanho	tempo médio(sec)
453	0.0000097861
826	0.0000010980
1397	0.0000011549
2278	0.0000011040
3762	0.0000010881
6151	0.0000010970
9523	0.0000013347

   (e) remoção

tamanho	tempo médio(sec)
453	0.0000316944
826	0.0000567167
1397	0.0001017833
2278	0.0001460733
3762	0.0002503188
6151	0.0003360525
9523	0.0005146545

Aqui pode-se notar um ganho de velocidade em todas as operações graças ao uso da busca binária. Embora nenhuma operação seja constante, como na tabela de hashing, no vetor ordenado todas as funções são logarítmicas, exceto pela inserção e remoção que são lineares, uma vez que temos de empurrar todo o vetor após inserir alguém em seu lugar(ou tirar). Logo, o vetor ordenado é uma estrutura de dados muito simples e relativamente rápida, sendo provavelmente a melhor opção para aplicações que não precisam de alta performance. Notamos que ao manter a ordem na inserção e remoção não é necessária nenhuma operação custosa de ordenação (Nlog n).

4. Lista Desordenada

   (a) inserção

tamanho	tempo médio(sec)
453	0.0000336546
826	0.0000657333
1397	0.0001127873
2278	0.0002182604
3762	0.0003479089
6151	0.0005675653
9523	0.0009609218

   (b) consulta

tamanho	tempo médio(sec)
453	0.0000361250
826	0.0000659363
1397	0.0001077853
2278	0.0002130158
3762	0.0003388287
6151	0.0005675931
9523	0.0009549050

   (c) rankeamento

tamanho	tempo médio(sec)
453	0.0000596639
826	0.0001096676
1397	0.0001881892
2278	0.0003108238
3762	0.0005213861
6151	0.0008472416
9523	0.0013299317

   (d) seleção

tamanho	tempo médio(sec)
453	0.0165136204
826	0.0597941127
1397	0.1667345069
2278	0.5298271505
3762	1.4323702564
6151	3.9749881297
9523	10.4804316990

   (e) remoção

tamanho	tempo médio(sec)
453	0.0000347880
826	0.0000661716
1397	0.0001109186
2278	0.0002199921
3762	0.0003486752
6151	0.0006416099
9523	0.0010614149

Na lista desordenada não há nenhum invariante que nos diga algo sobre a posição de qualquer nó. Portanto, todas as operações precisam percorrer toda a lista (lineares), exceto pela inserção que sempre insere no começo, e pela seleção que para cada nó percorre toda a lista (quadrática). Logo, embora seja um jeito rápido de guardar dados, ela não serve para muito mais do que isso. O caráter quadrático da seleção é muito marcante.

5. Lista Ordenada

   (a) inserção

tamanho	tempo médio(sec)
453	0.0000572870
826	0.0001087892
1397	0.0001743490
2278	0.0003315673
3762	0.0006407861
6151	0.0009949931
9523	0.0015872812

   (b) consulta

tamanho	tempo médio(sec)
453	0.0000553593
826	0.0001071902
1397	0.0001694333
2278	0.0003318050
3762	0.0006154802
6151	0.0009704574
9523	0.0014458574

   (c) rankeamento

tamanho	tempo médio(sec)
453	0.0000298352
826	0.0000611569
1397	0.0000955618
2278	0.0001849634
3762	0.0003391198
6151	0.0005111812
9523	0.0008058713

   (d) seleção

tamanho	tempo médio(sec)
453	0.0045118602
826	0.0162747529
1397	0.0405723069
2278	0.1468397109
3762	0.3996567921
6151	1.1323982921
9523	2.6395267030

(e) remoção

tamanho	tempo médio(sec)
453	0.0000498796
826	0.0001070814
1397	0.0001640265
2278	0.0003425683
3762	0.0006190317
6151	0.0012434297
9523	0.0016662822

Na lista ordenada não podemos tomar vantagem da busca binária, uma vez que as posições de memória não são indexadas. Contudo, há uma importante vantagem: A operação de seleção é linear, pois basta percorrer a lista e contar. Perde-se na inserção, que se torna linear, mas é um preço pequeno a pagar, em relação à lista desordenada.

6. Árvore binária

(a) inserção

tamanho	tempo médio(sec)
453	0.0000042139
826	0.0000037971
1397	0.0000046980
2278	0.0000048396
3762	0.0000061614
6151	0.0000051109
9523	0.0000054703

(b) consulta

tamanho	tempo médio(sec)
453	0.0000034815
826	0.0000038088
1397	0.0000040471
2278	0.0000056644
3762	0.0000043851
6151	0.0000042574
9523	0.0000062614

(c) rankeamento

tamanho	tempo médio(sec)
453	0.0000033565
826	0.0000059294
1397	0.0000042549
2278	0.0000051673
3762	0.0000042980
6151	0.0000059703
9523	0.0000045446

(d) seleção

tamanho	tempo médio(sec)
453	0.0000155259
826	0.0000196118
1397	0.0000247824
2278	0.0000213911
3762	0.0000249059
6151	0.0000257673
9523	0.0000270515

(e) remoção

tamanho	tempo médio(sec)
453	0.0000053509
826	0.0000050539
1397	0.0000058765
2278	0.0000073267
3762	0.0000053416
6151	0.0000055069
9523	0.0000058673

É gritante o ganho de desempenho com uma estrutura de dados que tem todas as operações no pior caso logarítmicas(seleção é $log^2 n$). Embora uma árvore, mesmo simples como essa, seja mais difícil de implementar do que uma estrutura linear, vemos que a aplicabilidade da divisão binária em todas as operações é extremamente desejável.

7. Treap

(a) inserção

tamanho	tempo médio(sec)
453	0.0000069528
826	0.0000091961
1397	0.0000078206
2278	0.0000099703
3762	0.0000097228
6151	0.0000090248
9523	0.0000132752

(b) consulta

tamanho	tempo médio(sec)
453	0.0000035769
826	0.0000040255
1397	0.0000042324
2278	0.0000045277
3762	0.0000045594
6151	0.0000070743
9523	0.0000056030

(c) rankeamento

tamanho	tempo médio(sec)
453	0.0000037352
826	0.0000039324
1397	0.0000043588
2278	0.0000063842
3762	0.0000048842
6151	0.0000070208
9523	0.0000059376

(d) seleção

tamanho	tempo médio(sec)
453	0.0000153917
826	0.0000204520
1397	0.0000227206
2278	0.0000267436
3762	0.0000261554
6151	0.0000334901
9523	0.0000412267

(e) remoção

tamanho	tempo médio(sec)
453	0.0000054046
826	0.0000053931
1397	0.0000058598
2278	0.0000060673
3762	0.0000063743
6151	0.0000082475
9523	0.0000088307

Os valores são semelhantes aos da árvore de busca binária. Os do treap parecem ligeiramente mais consistentes, mas parece que a quantidade de operações maior leva a tempos ligeiramente mais elevados, embora com menos variância.

8. Árvore 2-3

(a) inserção

tamanho	tempo médio(sec)
453	0.0000037019
826	0.0000040059
1397	0.0000043627
2278	0.0000055000
3762	0.0000048842
6151	0.0000049980
9523	0.0000054515

(b) consulta

tamanho	tempo médio(sec)
453	0.0000030481
826	0.0000028882
1397	0.0000031216
2278	0.0000033347
3762	0.0000034861
6151	0.0000035614
9523	0.0000039208

(c) rankeamento

tamanho	tempo médio(sec)
453	0.0000026120
826	0.0000027029
1397	0.0000029382
2278	0.0000033238
3762	0.0000033297
6151	0.0000033495
9523	0.0000043703

(d) seleção

tamanho	tempo médio(sec)
453	0.0000106093
826	0.0000119304
1397	0.0000158108
2278	0.0000156980
3762	0.0000182624
6151	0.0000182624
9523	0.0000224238

(e) remoção

tamanho	tempo médio(sec)
453	0.0000075000
826	0.0000083000
1397	0.0000091686
2278	0.0000098069
3762	0.0000106337
6151	0.0000111158
9523	0.0000115069

Podemos perceber que os valores da árvore 2-3 são muito rápidos, exceto pela remoção que é um pouco lenta, em compensação, pois o algoritmo é complexo. Entretanto, as outras operações compensam, graças à manutenção estrita do estado da árvore.

9. Árvore Rubro Negra

(a) inserção

tamanho	tempo médio(sec)
453	0.0000312545
826	0.0000297182
1397	0.0000277091
2278	0.0000341636
3762	0.0000483818
6151	0.0000280818
9523	0.0000194636

(b) consulta

tamanho	tempo médio(sec)
453	0.0000146273
826	0.0000243545
1397	0.0000142182
2278	0.0000158909
3762	0.0000255000
6151	0.0000151636
9523	0.0000162273

(c) rankeamento

tamanho	tempo médio(sec)
453	0.0000129000
826	0.0000283909
1397	0.0000144545
2278	0.0000169545
3762	0.0000167182
6151	0.0000153727
9523	0.0000227636

(d) seleção

tamanho	tempo médio(sec)
453	0.0000306273
826	0.0000521727
1397	0.0000300364
2278	0.0000528818
3762	0.0000590545
6151	0.0000537364
9523	0.0000463545

(e) remoção

tamanho	tempo médio(sec)
453	0.0000114909
826	0.0000131909
1397	0.0000113091
2278	0.0000115455
3762	0.0000131091
6151	0.0000122000
9523	0.0000123091

Em comparação com a 2-3, a rubro negra parece ter tempos com variação muito maior, embora sejam de magnitudes comparáveis. As árvores são tecnicamente equivalentes, logo tempos muito parecidos são esperados pois todas as operações tem a mesma velocidade.